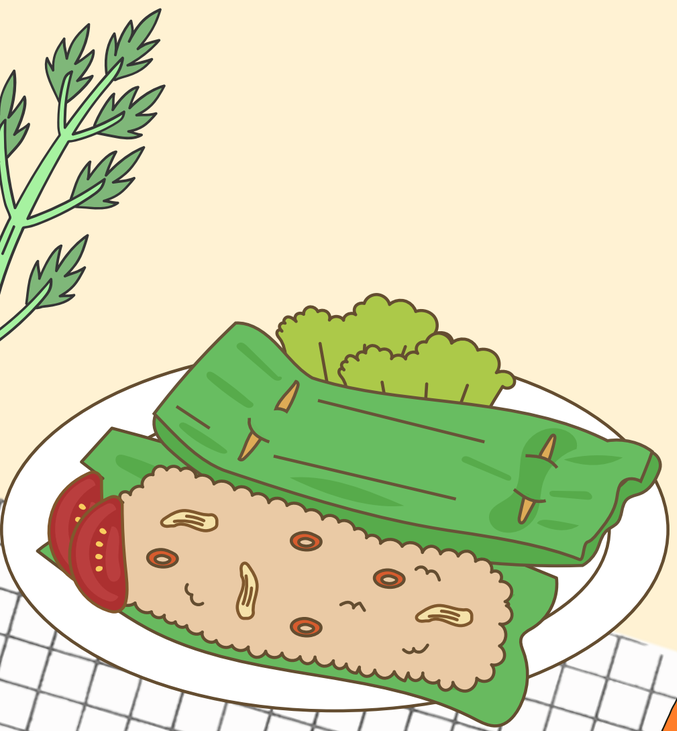
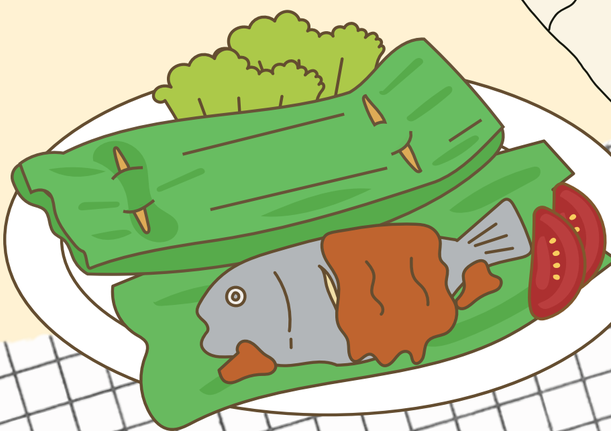
Nutrition Chatbot using Ollama (RAG + Fine-Tuning)

Smart Dietary Guidance



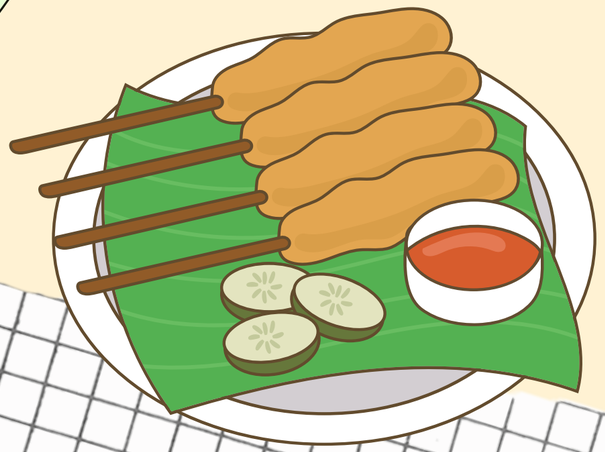


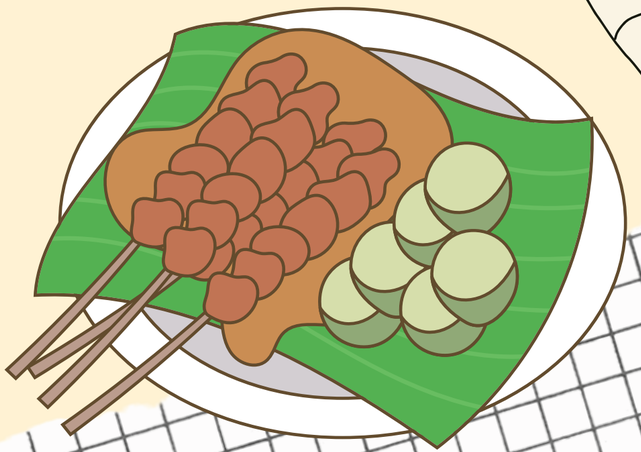
Group Member

Eiman Alrashdi

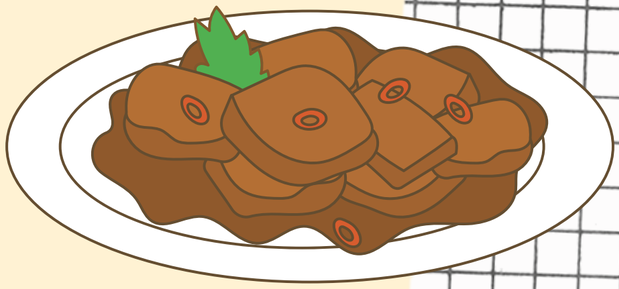
Nada AL-Ghassani

Ashgan ALshamali



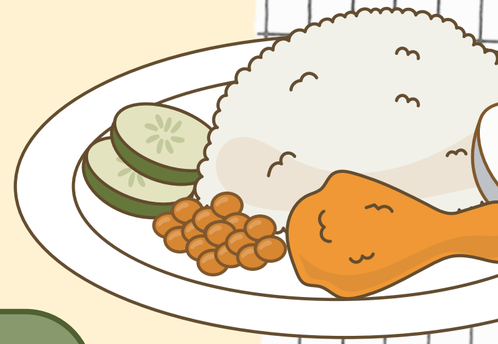


Introduction



Nutrition and healthy food are essential for a strong and active body. Healthy food provides important nutrients that help the body grow, stay energized, and prevent illness. Making good food choices supports overall health and well-being.

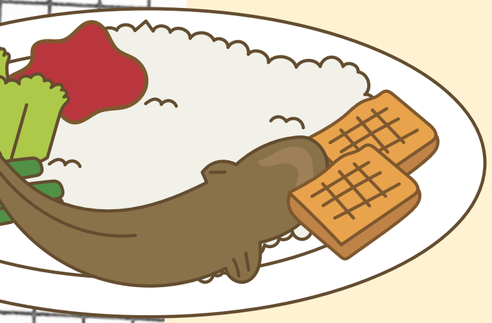
Problem Statement



1

- People struggle to understand nutrition
- Hard to find accurate food data
- Need for personalized, culturally relevant guidance

Gole



2

Build a Generative AI chatbot that provides nutrition insights using USDA Foundation Foods data.

Dataset Overview

1

Dataset Name:

- USDA Foundation Foods Dataset

2

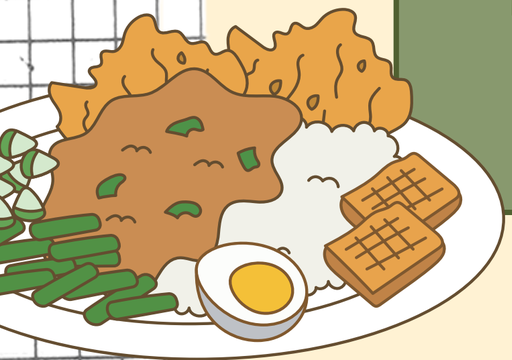
Size:

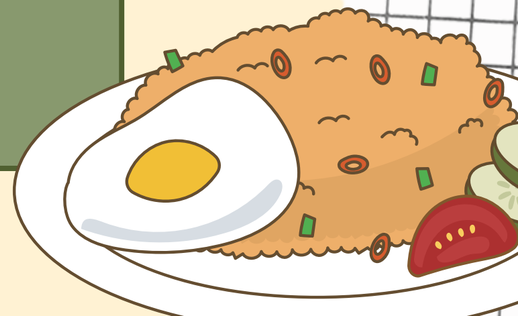
- 365 foods
- 150 nutrients per item
- JSON format

3

Why I Chose It:

- Accurate and trusted
- Very detailed nutrient information.
- Free and public.





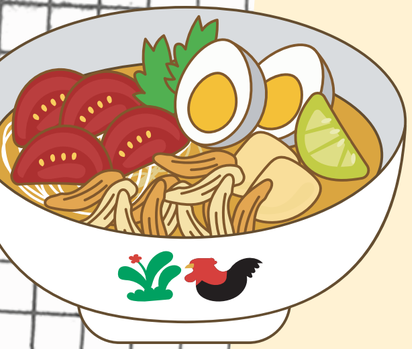
data collection

```python
from google.colab import files
1. # Upload the JSON file
uploaded = files.upload()  # FoodData_Central_foundation_food_json_2025-12-18.json

import json
2. # Get the uploaded filename
json_filename = next(iter(uploaded.keys()))
print("Loaded:", json_filename)

3. # Load JSON content
with open(json_filename, "r", encoding="utf-8") as f:
    data = json.load(f)

4.# Extract the FoundationFoods list
foods = data.get("FoundationFoods", [])
print("Total foods:", len(foods))
```

Choose Files FoodData_...5-12-18.json

FoodData_Central_foundation_food_json_2025-12-18.json(application/json) - 6801267 bytes, last modified: 1/12/2026 - 100% done

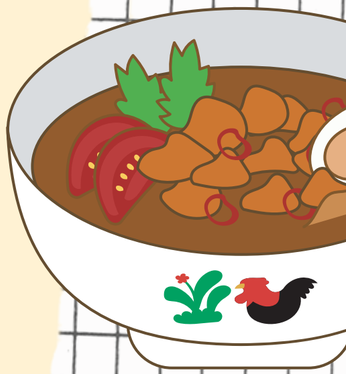
```
Saving FoodData_Central_foundation_food_json_2025-12-18.json to FoodData_Central_foundation_food_json_2025-12-18.json
Loaded: FoodData_Central_foundation_food_json_2025-12-18.json
Total foods: 365
```



Model Used:

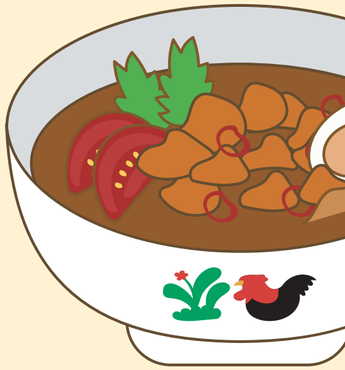
Training: Phi-3 Mini (4k Instruct, 4-bit) fine-tuned using QLoRA via Unsloth.

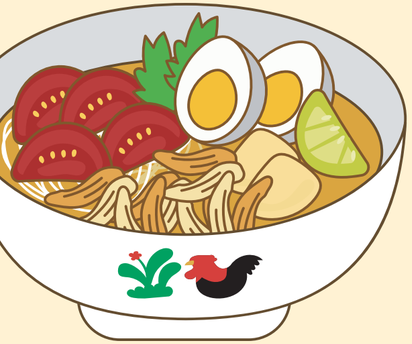
Inference / Deployment: LLaMA 3 deployed locally using Ollama with a RAG pipeline.



```python
# Import the FastLanguageModel class from Unsloth.
from unsloth import FastLanguageModel
# The name of the base model you want to fine-tune.
model_name = "unsloth/Phi-3-mini-4k-instruct-bnb-4bit"
# Maximum number of tokens the model can process in one input sequence.
max_seq_length = 2048
dtype = None

# Load the base model and tokenizer.
model, tokenizer = FastLanguageModel.from_pretrained(
    model_name=model_name,

    #maximum number of tokens
    max_seq_length=max_seq_length,
    dtype=dtype,
    # True makes training faster
    load_in_4bit=True,
    )
    # Convert the model into a PEFT (Parameter-Efficient Fine-Tuning) model using LoRA.
    model = FastLanguageModel.get_peft_model(
    #Pass the base model
    model,
    #rank of LoRA matrices
    r=16,
    target_modules=["q_proj","k_proj","v_proj","o_proj","gate_proj","up_proj","down_proj"],
    #scaling factor for LoRA updates
    lora_alpha=16,
    #to prevent overfitting
    lora_dropout=0.05,
    bias="none",
    use_gradient_checkpointing="unsloth",
    #ensures reproducible results
    random_state=42,
)
```

Loading the Base Model + Adding LoRA (QLoRA)





Fine-Tuning

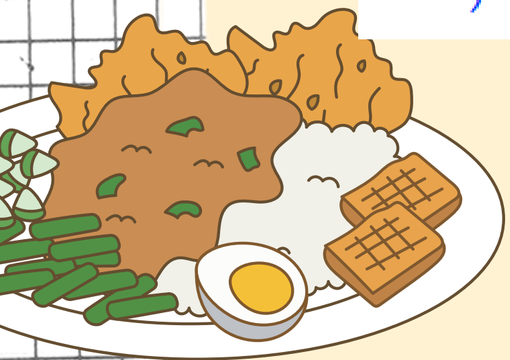
```python
from trl import SFTTrainer #(Supervised Fine-Tuning Trainer)
from transformers import TrainingArguments#(define all training hyperparameters.)

trainer = SFTTrainer(
    model=model,
    tokenizer=tokenizer,#convert text into tokens for the model
    train_dataset=dataset,#prepared for fine-tuning
    dataset_text_field="text",
    max_seq_length=2048,#Maximum number of tokens per training

    args=TrainingArguments(
        per_device_train_batch_size=2,
        gradient_accumulation_steps=4,
        warmup_steps=50, #increases learning rate for the first 50 steps
        learning_rate=2e-4, #learning rate for fine-tuning.
        num_train_epochs = 1, #model sees the entire dataset
        fp16=True, #faster training
        logging_steps=10,
        # Folder where training outputs and checkpoints will be saved.
        output_dir="nutrition_phi3_qlora",
        optim="adamw_8bit",
        weight_decay=0.01, #prevent overfitting
        lr_scheduler_type="linear",
        seed=42,
        report_to="none",
    ),
)
# Start the actual fine-tuning process.
trainer.train()
```

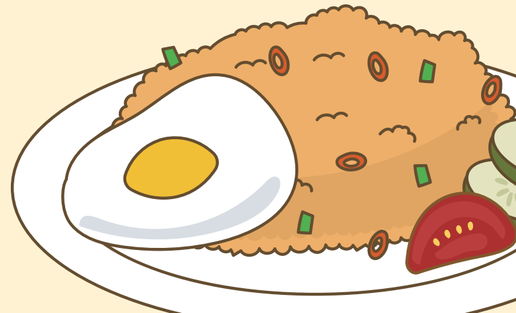
Saving the Fine-Tuned Model

```
# Save the LoRA fine-tuned model weights.
model.save_pretrained("nutrition_phi3_qlora_adapter")
# Save the tokenizer used during training.
tokenizer.save_pretrained("nutrition_phi3_qlora_adapter")
```

```
# Export model in GGUF format.
model.save_pretrained_gguf(
    "nutrition_phi3_gguf",
    tokenizer,
    quantization_method="q4_k_m" #use for real applications
)
```

export GGUF(GPT Generated Unified format)

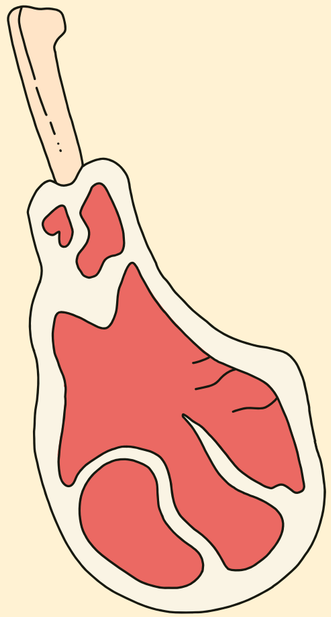




RAG

```python
def chunk_text(text: str, chunk_size: int = 900, overlap: int = 150) -> list[str]:
    text = re.sub(r"\s+", " ", (text or "")).strip()
    if not text:
        return []
    chunks = []
    start = 0
    while start < len(text):
        end = min(len(text), start + chunk_size)
        chunks.append(text[start:end])
        start = end - overlap
        if start < 0:
            start = 0
    return chunks
```

Split documents into chunks ✅



Generate embeddings ✅

```python
def build_upload_rag(file_obj):
    global UPLOAD_EMBEDDER, UPLOAD_INDEX, UPLOAD_TEXTS

    if file_obj is None:
        return "❌ أولًا ملف ارفعي. / Upload a file first."

    try:
        raw_text = read_uploaded_file(file_obj)
    except Exception as e:
        return f"❌ File read error: {e}"

    chunks = chunk_text(raw_text)

    if len(chunks) < 3:
        return "❌ استخراج فشل أو جدًا قصير النص PDF. جربي TXT أو مختلف ملف.\n❌ Too short / extraction failed."

    UPLOAD_EMBEDDER = SentenceTransformer("all-MiniLM-L6-v2")
    embs = UPLOAD_EMBEDDER.encode(chunks, show_progress_bar=False)
    embs = np.array(embs).astype("float32")

    UPLOAD_INDEX = faiss.IndexFlatL2(embs.shape[1])
    UPLOAD_INDEX.add(embs)
    UPLOAD_TEXTS = chunks

    return f"☑ Uploaded-file RAG ready! Chunks indexed: {len(UPLOAD_TEXTS)}"
```

RAG

Store them in a vector database (Faiss) ✅

```python
def build_rag_index(docs: list[str]):
    embedder = SentenceTransformer("all-MiniLM-L6-v2")
    embeddings = embedder.encode(docs, show_progress_bar=True)
    embeddings = np.array(embeddings).astype("float32")

    index = faiss.IndexFlatL2(embeddings.shape[1])
    index.add(embeddings)

    return embedder, index, docs

def retrieve_food_context(query: str, top_k: int = TOP_K) -> list[str]:
    q_emb = EMBEDDER.encode([query]).astype("float32")
    _, ids = FOOD_INDEX.search(q_emb, top_k)
    return [FOOD_TEXTS[i] for i in ids[0] if i != -1]


# ============================================================
```

```python
food_ctx = retrieve_food_context(message, top_k=TOP_K)
file_ctx = retrieve_upload_context(message, top_k=4)

rag_context = []
if food_ctx:
    rag_context.append("=== FoodData Central (Retrieved) ===")
    rag_context.extend(food_ctx)
if file_ctx:
    rag_context.append("=== Uploaded File (Retrieved) ===")
    rag_context.extend(file_ctx)
```

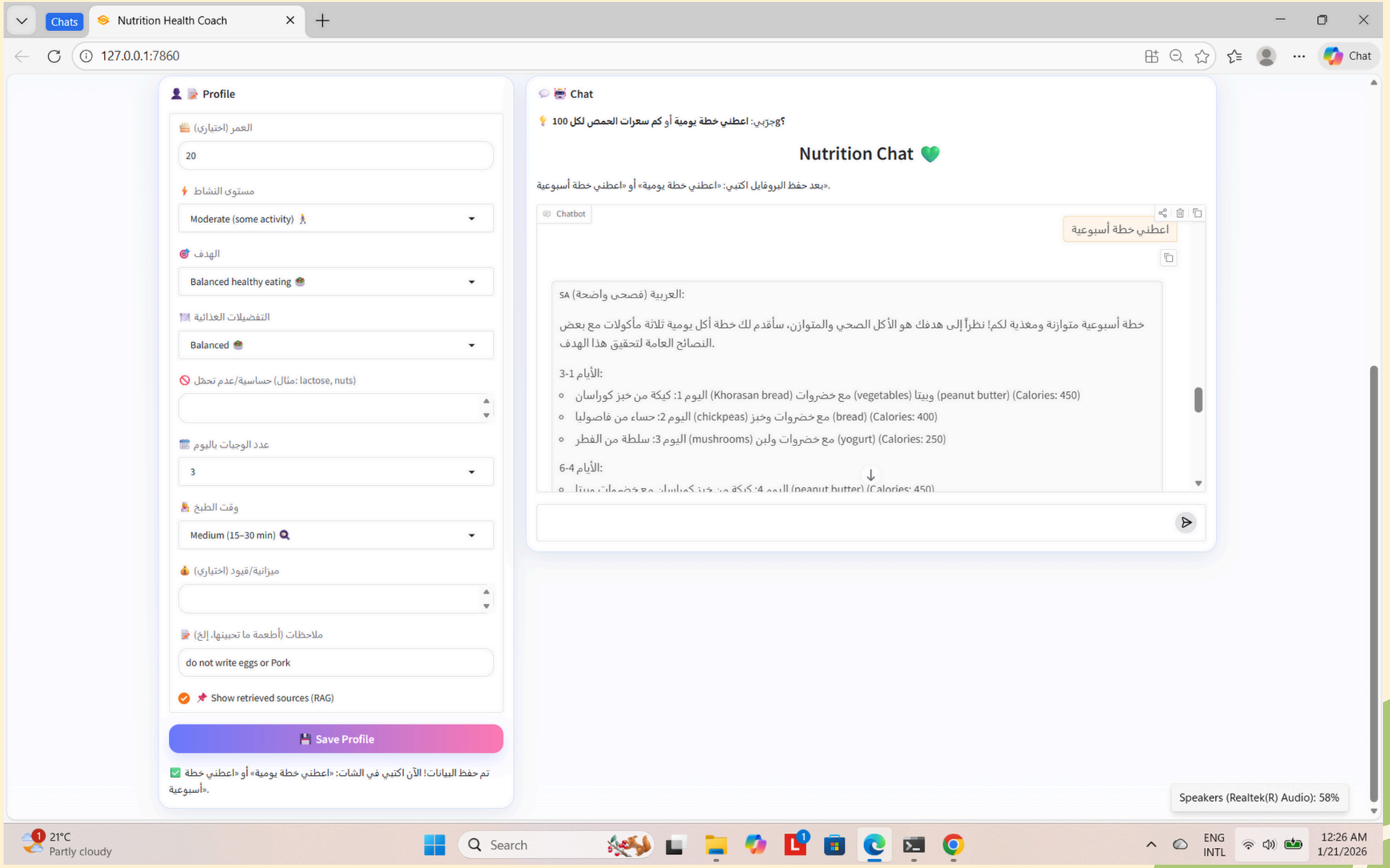
Retrieve relevant chunks during inference ✅

Interface

```
with gr.Blocks(title="Nutrition Health Coach", css=CUSTOM_CSS) as demo:

    with gr.Column(elem_id="title-card"):
        gr.Markdown(
            "## 🍏✨ **Nutrition Health Coach** (RAG + Ollama)\n"
            "مرحبًا! 👋 املئي البروفايل أولًا، ثم اسألي عن طعام 🍉 أو خطة اطلبي 🗓️ .\n\n"
            "<span id='pill'>📌 FoodData Central</span>"
            "<span id='pill'>📄 Upload TXT/PDF</span>"
            "<span id='pill'>🌍 AR + EN</span>\n\n"
            "☑️ **Important:** السعرات/الماكروز من FoodData فقط (تخمين بدون) 📝"
        )

    profile_state = gr.State({})

    with gr.Column(elem_id="section-card"):
        gr.Markdown("### 📄✨ **Upload File for RAG** (TXT / PDF)")
        if not PDF_OK:
            gr.Markdown("ℹ️ PDF غير متاح الآن. ثبّتي pypdf: `pip install pypdf` (TXT يعمل).")

        upload_file = gr.File(label="📤 Upload your file", file_types=[".txt", ".pdf"])
        build_rag_btn = gr.Button("🔧 Build RAG Index", variant="primary")
        build_rag_status = gr.Markdown("💡 الفهرس لبناء الزر اضغطي ثم الملف ارفعي.")

        build_rag_btn.click(fn=build_upload_rag, inputs=[upload_file], outputs=[build_rag_status])

    gr.Markdown("")
```

```
DISCLAIMER = (
    "⚠️ تنبيه: هذا الشاتبوت يقدّم إرشادات عامة للتغذية ونمط الحياة فقط ولا يقدّم نصائح طبية. "
    "إذا كان لديك أمراض/أعراض، أدوية/مكمّلات، فراجعي مختصًا.\n"
    "⚠️ Disclaimer: This chatbot provides general nutrition & lifestyle guidance only and does not provide medical advice. "
    "For medical conditions, medications, or symptoms, consult a qualified healthcare professional."
)
```

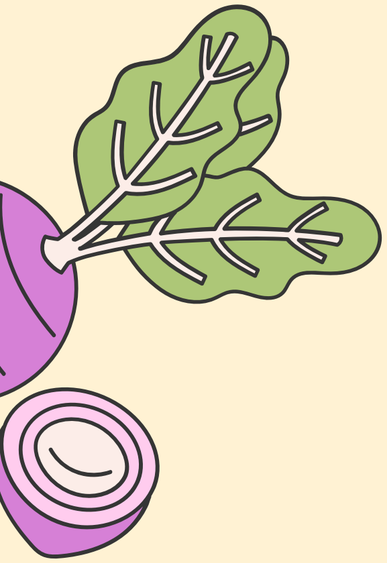
Interface


Chats
Nutrition Health Coach
127.0.0.1:7860
Chat
Profile
العمر (اختياري)
20
مستوى النشاط
Moderate (some activity)
الهدف
Balanced healthy eating
التفضيلات الغذائية
Balanced
حساسية/عدم تحمّل (مثال: lactose, nuts)
عدد الوجبات باليوم
3
وقت الطبخ
Medium (15–30 min)
ميزانية/قيود (اختياري)
ملاحظات (أطعمة ما تحبينها، إلخ)
do not write eggs or Pork
Show retrieved sources (RAG)
Save Profile
تم حفظ البيانات! الآن اكتبي في الشات: «اعطني خطة يومية» أو «اعطني خطة أسبوعية».
Chat
جرّبي: اعطني خطة يومية أو كم سعرات الحمص لكل 100g؟
Nutrition Chat
بعد حفظ البروفايل اكتبي: «اعطني خطة يومية» أو «اعطني خطة أسبوعية».
Chatbot
اعطني خطة أسبوعية
SA العربية (فصحى واضحة):
خطة أسبوعية متوازنة ومغذية لكم! نظراً إلى هدفك هو الأكل الصحي والمتوازن، سأقدم لك خطة أكل يومية ثلاثة مأكولات مع بعض النصائح العامة لتحقيق هذا الهدف.
1-3 الأيام:
اليوم 1: كيكة من خبز كوراسان (Khorasan bread) مع خضروات (vegetables) وبيتا (peanut butter) (Calories: 450)
اليوم 2: حساء من فاصوليا (chickpeas) مع خضروات وخبز (bread) (Calories: 400)
اليوم 3: سلطة من الفطر (mushrooms) مع خضروات ولبن (yogurt) (Calories: 250)
4-6 الأيام:
Speakers (Realtek(R) Audio): 58%
21°C
Partly cloudy
Search
ENG
INTL
12:26 AM
1/21/2026

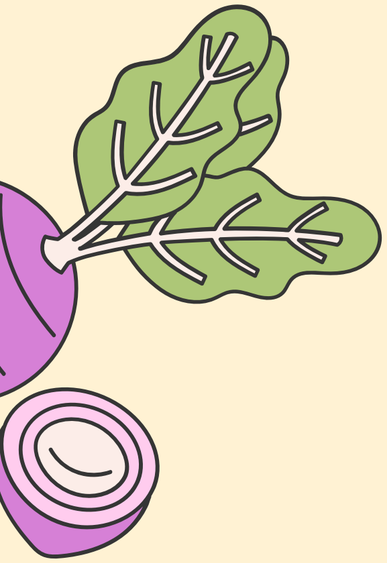
How are hallucinations reduced?

Hallucinations are reduced using multiple strategies:

- RAG grounding: The model is instructed to answer using only the retrieved documents.
- Explicit prompt rules: The prompt clearly forbids guessing or inventing numbers.
- Trusted data sources: Nutrition facts are retrieved exclusively from FoodData Central.
- Low temperature decoding: A low temperature is used to reduce randomness.
- Safety filtering: Medical questions are detected and refused instead of answered.



Ethical considerations



- The system includes a clear domain disclaimer stating that it does not provide medical advice.
- The chatbot avoids diagnosis, medications, or treatment recommendations.
- The responses are designed to be supportive, neutral, and non-harmful.

Thank You

For Your Attention